Introduction to Computer Organization – Fall 2023

Lab 2

Due: @11:55 PM, Wed September 13th via Gradescope

The following assignment is intended to be completed during your assigned lab period. One member of your group must submit the assignment to Gradescope by the posted deadline and indicate your group members when submitting the assignment. **Each group member must be present during the scheduled lab period in order to receive credit.**

NOTE: We have updated the way grades are calculated in labs to be more generous. See the syllabus for more details.

Group names and uniqnames

Pod Member Name	**Uniqname**
Yuzhen chen	yuzhench

For each of the following problems, one person should act as the "scribe" and log the discussions of the group. You should rotate who is the scribe for each problem and indicate in the given space.

Problem 1: Introduce Yourselves *[2 points, Completion]*

Scribe: **[Scribe's name here]**

This is the group of students you will be working with for the next few weeks. You will meet once a week to work through problems related to the class, but many students use these groups to form extra study groups to discuss projects, exams, etc.

Take a few minutes to go around the group and introduce yourselves.

Pod Member Name	**Year**	**What's something you are looking forward to this fall?**

Problem 2: Short answer *[12 points]*

Scribe: **[Scribe's name here]**

Indicate whether or not each of the following is a valid advantage of increasing the number of registers in an ISA, and give a brief (30 words or fewer) justification for each answer. Partial credit may be given for incorrect answers with well reasoned explanations.

1. Increasing the number of registers generally reduces the number of instructions needed to implement a C program. **[3]**

2. Increasing the number of registers reduces the number of memory accesses (i.e. loads and stores) needed to implement a program. **[3]**

3. Increasing the number of registers reduces the size of each instruction. **[3]**

4. Increasing the number of registers allows for more operands to be used in a single instruction. **[3]**

Problem 3: C-Code *[18 Points, Autograded]*

Implement the function `numHighBits` and submit to gradescope. It should return the number of bits that are set to one in the binary representation of the input `number`. You should use sizeof to determine how many bits the input is on the given system

```
	int numHighBits(int number);
```

Starter code: `wget eecs370.github.io/labs/lab2.tar.gz`

Problem 4: LC2K Assembler Test Cases *[18 points, Autograded]*

For this problem, your group must submit test cases that expose at least 3 of the instructor bugs in project 1a. Once you have written test cases that expose the bugs, you must write the corresponding correct machine code output and submit it to the autograder for full credit. In addition to the constraints listed in the project, each LC2K program you write must be limited to **5 lines** or fewer. No test cases should cause errors on a correct assembler. Each submission will be limited to 3 test cases, but fewer may be needed. Each output file name must be the same as the assembly, with the extension changed to .mc.

You are encouraged to use (and submit) these test cases if you are still working on P1a. A great strategy is to run these test cases on your assembler and "diff" your output with the correct output any time you make a change.